Stepwise understanding and execution of project- Short Video Analytics

# Data generator (Source datasets needed for the project)

* Based on the data dictionary shared, below is a Python framework to generate required data.

*/mnt/bigdatapgp/saurav/project/data\_generator/*

* Data Generator generates csv, json, xml data as per configuration done in input\_info.json in below directory .json data will be generated to kafka topic(real time) and xml and csv will be generated to the local linux path(batch data).

***/mnt/bigdatapgp/saurav****/project/data\_generator/resource/input\_info.json*

*To generate all types of data mention "****format\_of\_gen\_file****": ["json",”csv”,”xml”] in the input\_info.json file*

*Also, pls limit the* ***No\_of\_rows configuration value*** *based on volume of data needed.*

* To execute the python code, we have to run below after doing above settings-

***nohup python2 data\_gen\_execute.py &***

* Data Generated is stored in below base path()

/mnt/bigdatapgp/saurav/project/data\_generator/out/generate\_data/

* Below are the respective paths for Web(CSV), Mobile(XML) and Realtime App (JSON) data generated

***/mnt/bigdatapgp/saurav/project/data\_generator/out/generate\_data/2020-03-17/196/csv/data\_gen.csv***

***Sample-*** *liked,user\_id,video\_end\_type,minutes\_played,video\_id,geo\_cd,channel\_id,creator\_id,timestamp,disliked*

***For real time data create and pass the kafka topic name in line number 29 of data\_utility.py code of data generator***

***producer.send('jsonvideoanalytics',record\_write)***

***Sample data in kafka topic-***

*{"liked": true, "user\_id": 101, "video\_end\_type": "2", "minutes\_played": 3, "video\_id": 101, "geo\_cd": "EU", "channel\_id": 11, "creator\_id": 101, "timestamp": "09/01/2020 19:02:56", "disliked": true}*

***/mnt/bigdatapgp/saurav/project/data\_generator/out/generate\_data/2020-03-19/257/xml/data\_gen.xml***

***Sample-*** *<record><liked>True</liked><user\_id>101</user\_id><video\_end\_type>2</video\_end\_type><minutes\_played>3</minutes\_played><video\_id>101</video\_id><geo\_cd>EU</geo\_cd><channel\_id>11</channel\_id><creator\_id>101</c*

*reator\_id><timestamp>09/01/2020 19:02:56</timestamp><disliked>True</disliked></record>*

* *Lookup data is created manually as per data dictionary in the below directory*

*/mnt/bigdatapgp/saurav/project/Lookup data/*

***Following are the files***

*channel-geocd.txt user-creator.txt user-subscn.txt video-creator.txt*

# Create and Populate Lookup tables in Hive and Hbase (One time work)

Creating and populating hive tables on top of hbase tables for data enrichment and filtering.

* Run the below scripts to create and load hbase tables-

**sh /mnt/bigdatapgp/saurav/project/code/populate-hbase-lookup-tables.sh**

Check if the data is populated correctly or nor

Start **hbase-shell**

>scan 'channel-geo-map'

>scan 'subscribed-users'

>scan 'video-creator-map'

* Run the below script to create and load hive-hbase tables

**hive -f /mnt/bigdatapgp/saurav/project/code/populate\_hive\_lookup.hql**

Check if all the tables are populated correctly-

select \* from channel\_geo\_map;

select \* from subscribed\_users;

select \* from video\_creator\_map;

# Create data tables in Hive (One time work)

There are three kinds of tables loaded using this approach created by below script-

* Initial landing table - **Merged\_data\_all**
* Intermediate enriched table - **enriched\_data**
* Final analysis tables- **top\_channels,users\_behaviour,connected\_creators,top\_royalty\_videos,top\_unsubscribed\_users**

**hive -f /mnt/bigdatapgp/saurav/project/code/create\_data\_tables.hql**

**hive -f /mnt/bigdatapgp/saurav/project/code/create\_final\_results.hql**

# Populate data tables (Batch process)- Spark

* **Load data to HDFS**

hdfs dfs -put /mnt/bigdatapgp/saurav/project/data\_generator/out/generate\_data/2020-03-17/196/csv/data\_gen.csv /tmp/

hdfs dfs -put /mnt/bigdatapgp/saurav/project/data\_generator/out/generate\_data/2020-03-19/257/xml/data\_gen.xml /tmp/

* **Parse,load and merge data to Hive landing table**

cat '/mnt/bigdatapgp/saurav/project/code/load\_data.scala’ | spark2-shell --num-executors 18 --executor-cores 3 --executor-memory 3g --jars /mnt/bigdatapgp/saurav/project/spark-xml\_2.11-0.4.1.jar

* **Load data to Enriched table**

hive -f /mnt/bigdatapgp/saurav/project/code/enrich\_data.hql

* **Load data to Final analysis tables**

hive -f /mnt/bigdatapgp/saurav/project/code/load\_final\_results.hql

# Populate data tables (Stream process)- Spark

Below Spark-submit command can stream data into HDFS after getting it from kafka , exploding the json by parsing it and then writing the data in HDFS after enriching it

spark-structured streaming code which will get data from a kafka topic and write it to a HDFS path after doing a lookup on data stored in HDFS path.

**Code Source:** https://drive.google.com/file/d/1sHcckx5FmcJWQI6z8kH1aIrx4P\_XQMfj/view?usp=sharing

export SPARK\_KAFKA\_VERSION=0.10

spark2-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.11:2.1.0,org.apache.kafka:kafka-clients:0.10.2.1 --class com.saurav.training.streamtoHDFS --master yarn /mnt/bigdatapgp/ed

ureka\_510636/project/training-0.0.1-SNAPSHOT-jar-with-dependencies\_parq.jar